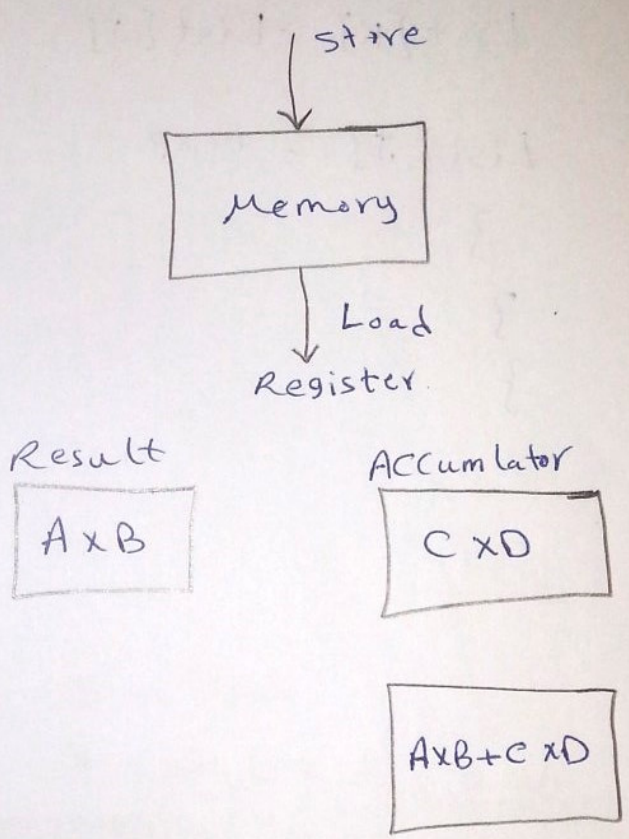
كارة حاسب

sheet 3

٢/٤

2 $A \times B + C \times D$

Load A
Multiply B
store Result
Load C
Multiply D
ADD Result
store result



3

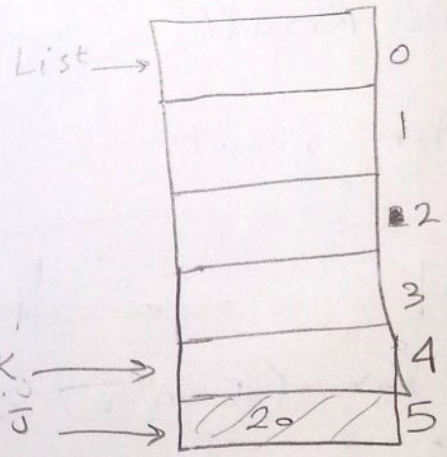
```
for (J=n-1 ; J>=0 ; J--)
{
for (K=J-1 ; K>=0 ; K--)
{
if (List[K] > List[J])
{
```

```
{
temp =List [K];
List[K] =List[J];
List[J] = temp
  }
 }
}
```

Note

ك ← ثابت

K ← يتحرك لأعلى ويرى هل القيمة التى يشير اليها أكبر من ك ولا لأ وعلى أساس ده هيعيد الترتيب.



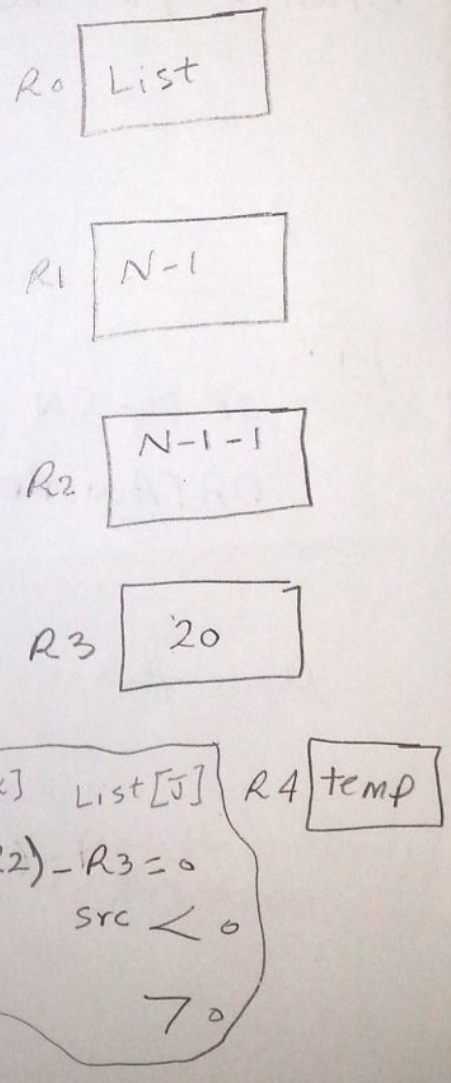
← بعد الإنتهاء من ال (Loop) ننتقل لل (Loop) جديدة فنقوم بتحريك ك فى مكان آخر ونثبتها. ويحرك K كما سبق.

سـ رقم (٣) هو مثال لـ (insertion sort)

سـ السؤال عايز برنامج (assembly) يمثل الـ (Code).

Sol

```
        Move  #List, R0
        Move  N, R1
        Subtract #1, R1
out     Move  R1, R2
        Subtract #1, R2
        Movebyte (R0,R1), R3
Inner   Comparebyte R3, (R0,R2)
        Branch <=0  Next
        Movebyte (R0,R2), R4
        Move byte R3, (R0,R2)
        Movebyte R4, (R0,R1)
```



Next Decrement R2

Branch ≥ 0 Inner

Decrement R1

Branch ≥ 0 out.

معناها لو نتيجة المقارنة أكبر من الصفر هنرجع لأول ال (loop) اللى إسمها (Inner)

---

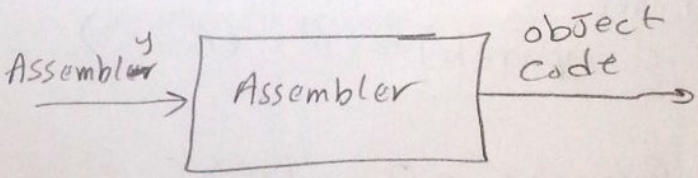
4)

ORIGIN 1000

DATAWORD 300

Move #300, 1000

- الإثنان بيقوموا بنفس الوظيفة لكن في أوقات مختلفة.
- الأول Assembler Directives



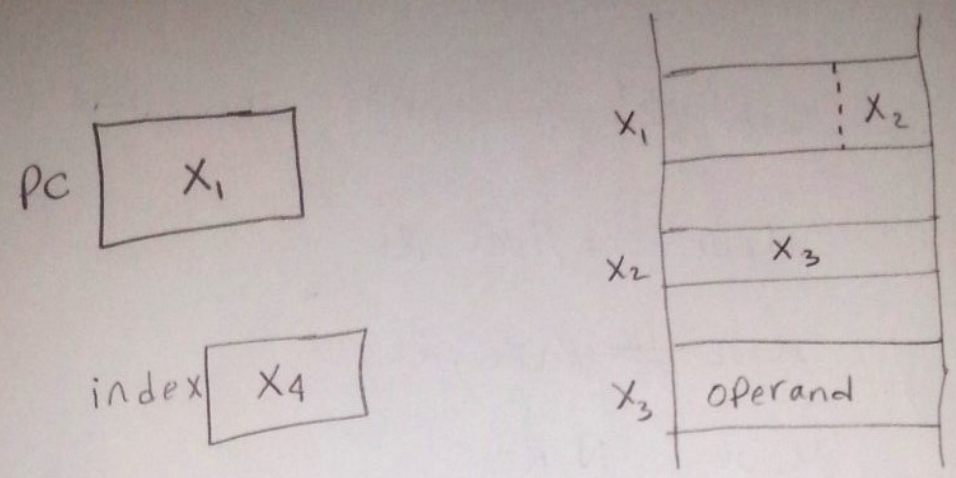
- معلومات إضافية لـ (Assembler) عشان ينفذ الـ Code.

5

عامل زى ال (Subroutine) او (Function) البرمجة.

```
        Move  #AVEC, R1
        Move  #BVEC, R2
        Move  N, R3
        Call  Sub
        Move  R0, DOTPROD
Sub     CLR   R0
loop    Move  (R1)+, R4
        Multiply (R2)+, R4
        ADD   R4, R0
        Decrement R3
        Branch To loop
        Return
```

ميتفعش (loop) تبدأ منه عند (CLR) لأنها كده هتلغى الناتج اللى كتباه فى R0.



في الحالات الآتية ماذا سيحدث.

Direct ⟹ $X_3 = X_2$

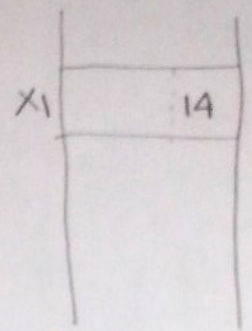
indirect ⟹ $X_3 = (X_2)$

Pc relative ⟹ $X_3 = X_2 + X_1 + 1$

(PC) تزيد بـ 1 عشانه يشاور على الأمر التالي.

indexed ⟹ $X_3 = X_2 + X_4$

7



ده اللى تغير فى أدلبرامج.
عنوانه
عايز يعرف محتواه ال (operand)

immediate ← operand is 14.

Move #14, R1
ينقل (14) فى R1 محتواه.

Direct ⊃ 14

Move 14, R1
ده معناه انك عنوانه
هو نقله فى R1.

indirect

in memory location whose address is 14

register

register indirect

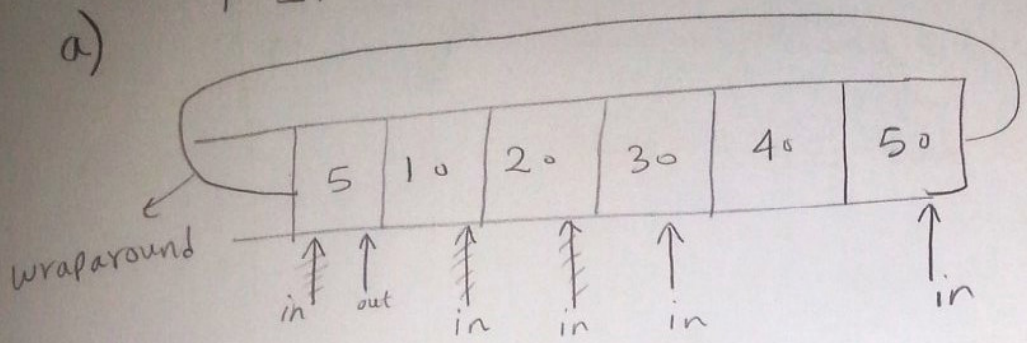
14 ده عنوان register
register ده فيه العنوان

memory Loc. whose address in register 14.

8

FIFO

a)



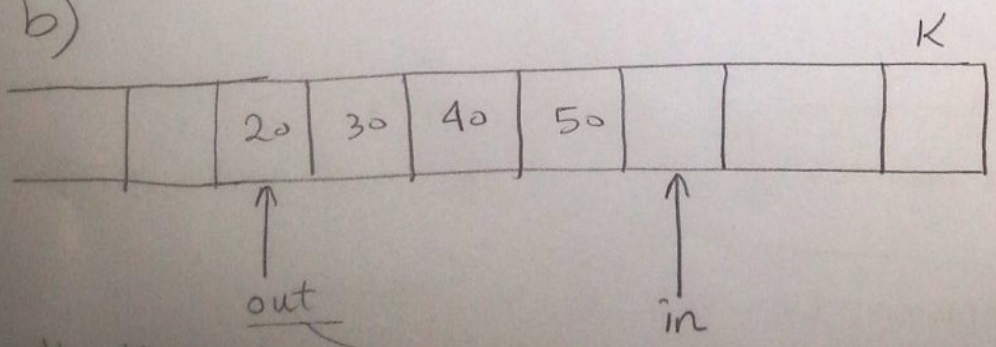
- Queue ← (in) بيتحرك جوه ال

← (out) لتحريكه من 5 ل 10 نقوم بحذف 5 من ال Queue فيشير لـ 10.

← هيحط نقط لحد ما يوصل ل (50) وبيقا معيش مكان تاني هيلف بقا ويعمل (wrap around)

ده حل السؤال (a)

b)



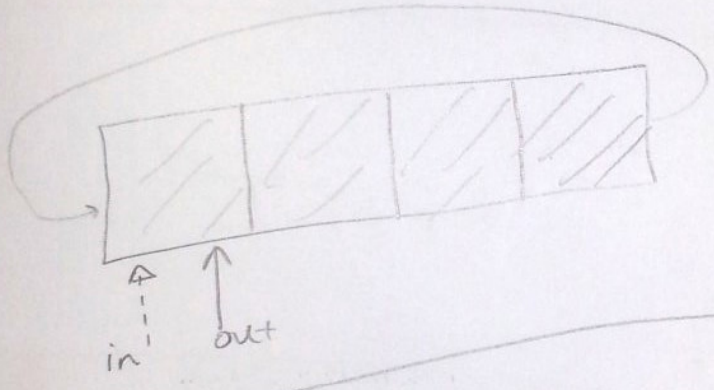
out ← يشير على القيمة اللي هتتحذف

c)

الحل

* عندما تكون (In = out) يشاور واعلى تفسيرين ممكنين
وهيكون (Queue) ده فاضي.

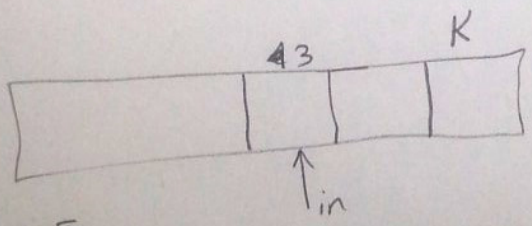
* أو (in) لسه قرب على ملء الأماكن الفارغة وعمل (wrap around)
ووجد له (Queue) ممتلئ فلسه يقوم بشيء.



d) (in + 1) mod K

قيم K

ex



if in = 3, K = 5

4 mod 5 = 4

طالما القيمة أقل من K
الناتج يساويها.

d)

Loc ← in

out ← (in+1) mod K

← لو تساوى (out) هنقول فيه ال (operation)

← مش متأكد منها.

٩

REPORT

sheet 4

1 WMFC

2

1- PCout, MARin, Read, select 4, Add, Zin

2- Zout, PCin, Xin, WMFC

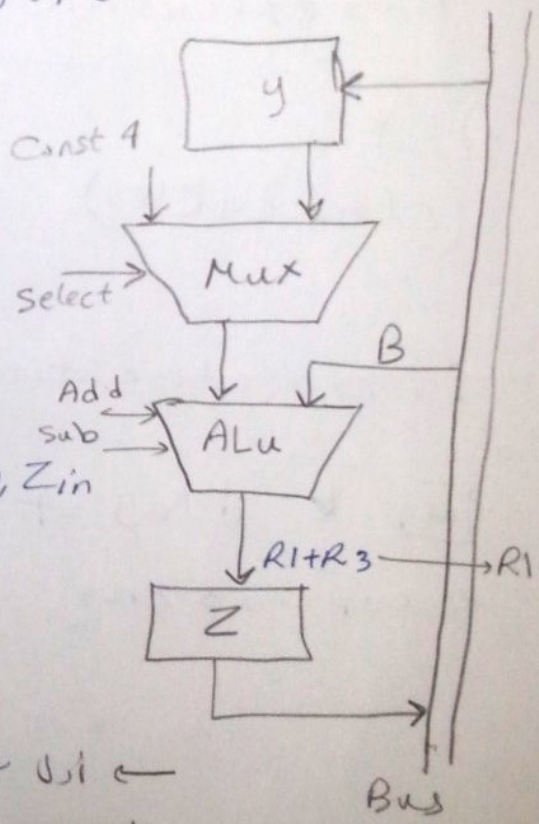
3- MDRout, IRin

4- R3out, MARin, Read

5- R1out, Xin, WMFC

6- MDRout, select Y, ADD, Zin

7- Zout, R1in, End



(Fetch) ← اول ٣ اوامر هما أوامر الـ
ثابتين فى كل البرامج.

A ADD (R3), R1

← هو (indirect) بسبب خطوة رقم (٤) لأننا

خرجنا محتواه على (MAR).

← هما ٧ خطوات فإذا هياخدوا

7 clock cycles

↳ execution time

3

WMFC ← بأخذ ونحقق عدد ال

(clock cycles)

∴ execution time = 9 clock cycle

---

4 delay ⇒ Bus = 0.3 ns , ALU = 2 ns

setup → 0.2 ns ; Hold time → 0

أقل وقت تاخذ هتجمع 2 ، 0.3 ، 0.2

time = 2.5 ns